
CSCI 567 – JYM

Final Project Report

James Christiansen-Salameh
University of Southern California
Los Angeles, CA, 90007
jameschr@usc.edu

Yuanqing Li
University of Southern California
Los Angeles, CA, 90007
yli96854@usc.edu

Maritsa Negrete
University of Southern California
Los Angeles, CA, 90007
negretem@usc.edu

Abstract

Our team trained an XGBoost model to predict the 35th month sales for a test set of shop and item combinations for Kaggle's Predict Future Sales competition. Training data consisted of sales records from months leading up to the 35th month. We achieved a RMSE of .89273 on the testing set. We also experimented with an ensemble of XGBoost, CatBoost, and LightGBM but found that the ensemble did not outperform XGBoost alone.

1 Introduction

This project involved training a machine learning model to predict the sales for given shop and item combinations in a given month for Kaggle's Predict Future Sales data science competition. [1] The Kaggle dataset contained data for sales for various shop and item combinations for 34 months, and our task was to predict the sales for the test dataset for the 35th month. We approached this task by first exploring the given dataset to determine the features to be used in our machine learning algorithm. We tried out three different algorithms before choosing to focus on XGBoost, and then focused on optimizing XGBoost's hyperparameters to improve our root-mean-square error (RMSE).

2 Previous Work

We found inspiration for data analysis, feature engineering, gradient boosting, and general sales prediction strategies from a variety of sources. Source [7] gave us general direction regarding data cleaning, feature engineering, and train/test set splitting. Source [3] helped us with the specifics of feature engineering and tuning model parameters. Source [6] gave a great overview of the three gradient boosting models we used, and tips for improving the model training process. Source [2] went into great detail regarding the design and functionality of XGBoost. Source [1] described a successful strategy for a similar sales prediction contest.

3 Data

We chose `item_cnt_day` as the target variable. The following describes how we preprocessed and training data and extracted features of interest.

3.1 Data analysis and processing

The quality of the training data drives the accuracy of the model. We analyzed the training data to find unreasonable data and remove or modify it.

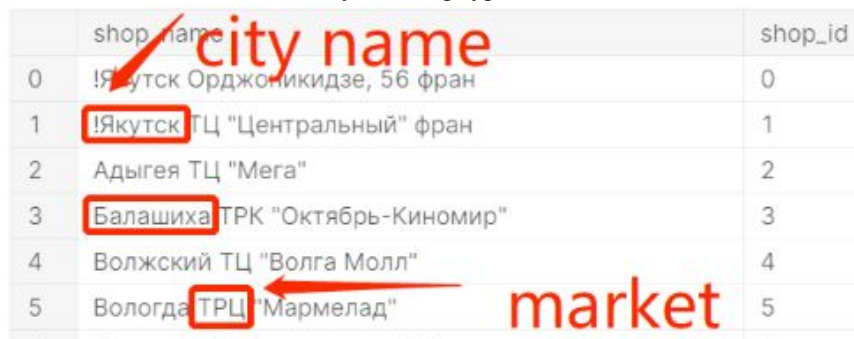
3.1.1 For the data in `sales_train.csv`, we examined the highest and lowest prices for all of the items. We found the highest price to be 307980.0 while the lowest price was -1. For the `item_price` field, we found it unreasonable for an item to have a price with a negative value or value of 0, so cleared the data with a price less than 1.

3.1.2 For the `item_cnt_day` field, we found the highest value for daily sales to be 2169 while the lowest value was -22. We didn't think it made sense for the daily sales of goods to be less than 0, so we replaced any negative values with 0.

3.2 Feature engineering

3.2.1 Shop

The shop file has two fields, the name of the item and the id of the item. The name of the product is in Russian, which is not easy to understand, but we used Google translate to look at the shop names in English and noticed a pattern. For a shop name, the first word is usually the city name, and the second word is usually one of a few words that are repeated throughout the shop names, suggesting it is a categorical label for the shops. Therefore, it is possible to separate the product name and build two features, city and shop type.



	shop_name	shop_id
0	Иркутск Орджоникидзе, 56 фран	0
1	Иркутск ТЦ "Центральный" фран	1
2	Адыгея ТЦ "Мега"	2
3	Балашиха ТРК "Октябрь-Киномир"	3
4	Волжский ТЦ "Волга Молл"	4
5	Вологда ТРЦ "Мармелад"	5

3.2.2 Categories

The item categories file has two fields, `item_category_name` and `item_category_id`. There are two features that can be extracted from `item_category_name`, a major category and second, more specific, category, which are usually separated by a hyphen.

	item_category_name	item_category_id	type_code
0	PC - Гарнитуры/Наушники	0	PC
1	Аксессуары - PS2	1	Игры
2	Аксессуары - PS3	2	Игры
3	Аксессуары - PS4	3	Игры
4	Аксессуары - PSP	4	Игры

Major categories of game consoles

Secondary category of game consoles

3.2.3 items

	item_name	item_id	item_category_id
0	! ВО ВЛАСТИ НАВАЖДЕНИЯ (ПЛАСТ.) D	0	40
1	IABBY FineReader 12 Professional Edition Full...	1	76
2	***В ЛУЧАХ СЛАВЫ (UNV) D	2	40
3	***ГОЛУБАЯ ВОЛНА (Univ) D	3	40
4	***КОРОБКА (СТЕКЛО) D	4	40
...
22165	Ядерный титбит 2 [PC, Цифровая версия]	22165	31
22166	Язык запросов 1С:Предприятия 8 [Цифровая версия]	22166	54
22167	Язык запросов 1С:Предприятия 8 (+CD). Хрустале...	22167	49
22168	Яйцо для Little Inu	22168	62
22169	Яйцо дракона (Игра престолов)	22169	69

22170 rows × 3 columns

For items, take the content inside “()” as a feature, and the content inside “[]” as a feature.

3.3 Construction of time-series features

For the monthly sales forecast, the past sales data is very important, the sales data of the past month, the sales data of the past two months, the sales data of the past three months, the average monthly sales data of the past, etc.

Similarly, such features include:

- Monthly sales of each item for each store over the past one, two, three months.
- Every month, all stores sell an average of all goods.
- Last month, the average sales of all goods in all stores (can reflect the sales trend).
- The average number of items sold in all stores each month, and the number sold in the first two to three months.
- The average number of items sold in a store each month, and the number of items sold in the first two to three months.
- The average number of items sold in a store each month, and the sales volume in the first three months.
- The average number of items sold per store per month, per item category, and the previous month's sales.
- The average number of goods sold in each city each month, as well as the previous month's sales.
- Average monthly sales per item per city per month, and sales in the previous month.
- The average selling price of each commodity for each month, and the selling price for the first three months, and the price difference (reflecting the commodity price change trend).

- K. The turnover per store per month, and the ratio of the turnover to the average turnover for the month .
- L. Month characteristics, goods may appear seasonal changes in sales.
- M. Day characteristics, goods may appear within the month sales fluctuations.
- N. When the product is first sold, the new product may surge in the first sale, and the popularity will gradually decrease as the time goes by.

4 Experiments

4.1 Models

Our team’s best performing model and learning algorithm is XGBoost which is one of the most popular machine learning techniques. First, we used feature engineering to capture time-series trends and it lags from one month to three months. Second, we used validation-based early stopping to prevent overfitting. To be specific, the training set covers the first to thirty-second months and the thirty-third months used as a validation set.

XGBoost is used for supervised learning problems. It belongs to the category of boosting algorithms in 3 commonly used integration methods (bagging, boosting, stacking). It is an additional model. The base model generally selects the tree model, but other types of models such as logistic regression can also be selected. The idea of the algorithm is to continuously add trees and continuously split the features to grow a tree. Each time you add a tree, you learn a new function to fit the last predicted residual. When we finish training to get k trees, we have to predict the score of a sample. In fact, according to the characteristics of this sample, a corresponding leaf node will fall in each tree, and each leaf node corresponds to a score. The score corresponding to each tree needs to be added up to be the predicted value of the sample.

$$\hat{y} = \phi(x_i) = \sum_{k=1}^K f_k(x_i)$$

The reason why we choose XGBoost is that we checked many online materials and found that XGBoost is an algorithm that often appears in competition. According to Tianqi Chen & Carlos Guestrin (2016), “The impact of the system has been widely recognized in a number of machine learning and data mining challenges. Take the challenges hosted by the machine learning competition site Kaggle for example. Among the 29 challenge winning solutions 3 published at Kaggle’s blog during 2015, 17 solutions used XGBoost. Among these solutions, eight solely used XGBoost to train the model, while most others combined XGBoost with neural nets in ensembles. For comparison, the second most popular method, deep neural nets, was used in 11 solutions. The success of the system was also witnessed in KDDCup 2015, where XGBoost was used by every winning team in the top-10. Moreover, the winning teams reported that ensemble methods outperform a well-configured XGBoost by only a small amount.” (Tianqi Chen & Carlos Guestrin, 2016), we can see XGBoost works well in Kaggle competition.

Our team tried three different ML models: XGBoost, LightGBM, and Catboost. Each of these models is an implementation of gradient boosting. (add description + discussion of parameters for each)

RMSE of each single model

	XGBoost	LightGBM	Catboost
RMSE	0.89273	0.90692	1.25006

There are many ways to merge 3 models together, such as stacking, blending and so on. I took the average of 3 models. (Xgboost + lightgbm + catboost) / 3)

An ensemble of XGBoost, LightGBM and Catboost

	Ensemble
RMSE	0.8990

However, the average of the 3 models had a higher RMSE than XGBoost alone. We experimented with different weights for the ensemble average, but did not come up with a weighting that performed better than XGBoost alone. Tuning the CatBoost or LightGBM hyperparameters further may have led to a better ensemble result, but we decided to shift our focus to XGBoost alone.

Finally, we got our team's best value of RMSE (root-mean-square error) which is 0.89273.

In addition, we used some parametric models:

- max_depth - Maximum tree depth for base learners
- n_estimators - Number of gradient boosted trees
- min_child_weight - Minimum sum of instance weight(hessian) needed in a child
- colsample_bytree - Subsample ratio of columns when constructing each tree
- Subsample - Subsample ratio of the training instance
- random_state - Random number seed(Random_state is for reproducible results)

4.2 Evaluation

The data is shared into the training set and validation set. The validation set has never been seen in the training process of the model, so the performance on the validation set is used as the performance of the generalization ability, and the performance on the validation set is the local evaluation of the model performance. Our validation set included all data from month 33 and our training set included all other data from months 4 to 32. Data points from months 1 to 3 were dropped, seeing as our time series data could not be accurately calculated for these months. Test data was all from month 34, and was evaluated by RMSE after upload to Kaggle. We used month 33 as the validation set because we believed the data in month 33 would be most similar to month 34, compared to using a randomly distributed validation set. In all models, RMSE was used to calculate error between predicted price and actual price. Our XGBoost and LightGBM models used early stopping to prevent overfitting, meaning that when the validation set error did not improve for a certain number of rounds the training was stopped.

4.3 Setup

All models were implemented in the Python programming language with xgboost, catboost, and lightgbm libraries. The pandas library was used to process, store, and manipulate data from the original csv file format. The numpy library was used for matrix manipulation processes. The pickle library was used for object serialization.

Our computational resources were limited. Most training and prediction was executed on laptops with 16GB of RAM. One team member used Google's Colaboratory, which provides computing resources in the form of Jupyter notebooks where one can write and execute code. Running the code in a Colab notebook was faster than running on a computer with limited RAM, and it allowed us to combine code with graphical outputs to easily explore the data and share our findings.

5 Results

What results did you get?

Our best model, XGBoost, had an RMSE, or mean square error between the predicted value of the model and the real value, of 0.89273.

There is no obvious overfitting in my model from the following two points :

1. In the process of training the model, I set up early stopping. Specifically, if the performance of the model in 20 continuous iterations does not improve on the verification set, then stop training to prevent overfitting.
2. From the results, the RMSE of the optimal model on the training set was 0.81992, and the performance on the verification set was 0.91519. The model was slightly overfitted, but basically on the same order of magnitude.

```
[47] validation_0-rmse:0.81140 validation_1-rmse:0.90942
[28] validation_0-rmse:0.80868 validation_1-rmse:0.90993
[29] validation_0-rmse:0.80688 validation_1-rmse:0.90995
[30] validation_0-rmse:0.80415 validation_1-rmse:0.91164
[31] validation_0-rmse:0.80186 validation_1-rmse:0.91168
[32] validation_0-rmse:0.80013 validation_1-rmse:0.91246
[33] validation_0-rmse:0.79832 validation_1-rmse:0.91229
[34] validation_0-rmse:0.79651 validation_1-rmse:0.91267
[35] validation_0-rmse:0.79450 validation_1-rmse:0.91370
[36] validation_0-rmse:0.79349 validation_1-rmse:0.91394
[37] validation_0-rmse:0.79200 validation_1-rmse:0.91449
[38] validation_0-rmse:0.79100 validation_1-rmse:0.91470
[39] validation_0-rmse:0.78937 validation_1-rmse:0.91449
[40] validation_0-rmse:0.78831 validation_1-rmse:0.91458
[41] validation_0-rmse:0.78706 validation_1-rmse:0.91461
[42] validation_0-rmse:0.78551 validation_1-rmse:0.91470
[43] validation_0-rmse:0.78443 validation_1-rmse:0.91475
[44] validation_0-rmse:0.78348 validation_1-rmse:0.91519
Stopping. Best iteration:
[24] validation_0-rmse:0.81992 validation_1-rmse:0.90707

[1] training's l2: 1.4793 valid_1's l2: 1.28536
```

In addition, there is little difference between the performance of the validation set 0.91519 and that of the test set 0.89273, indicating that the validation set can well represent the generalization ability of the model without overfitting.

6 Conclusion

The performance of the optimal model in the top 5% indicates that the model constructed is effective.

Why it works:

1. Data cleaning. Data cleansing ensures that there is no illogical data, such as prices that cannot be negative. A clean data model is a valid premise. Even the best model won't work.
2. Feature engineering. Building the basic characteristics and business-aligned characteristics is the cornerstone of model excellence. Excellent features can make the model better approach the on-line data. More importantly, excellent features enable the model to learn the underlying laws of the data, rather than fitting the specificity of the data.
3. In the process of model construction, early stopping technology is used, which can effectively alleviate overfitting and make the model maintain the ability of generalization.
4. From the results, the model offline verification set is close to the test set, which also confirms the effectiveness of the model.

Different attempts:

1. For data preprocessing, more diverse processing methods can be considered. In this scheme, data with negative price is deleted. You can think of filling as mean or something like that.
2. Try more elaborate feature engineering.
3. Adopt more integrated learning methods for result fusion.

References

- [1] AI Institute. Kaggle Commodity Sales Forecast 3rd Place Plan Released. Online, 2018. URL <https://cloud.tencent.com/developer/article/1166628>
- [2] Chen, Tianqi, and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. 10 June 2016. arxiv.org/pdf/1603.02754.pdf.
- [3] Kaggle. Feature engineering, xgboost. Online, 2018. URL <https://www.kaggle.com/dlarionov/feature-engineering-xgboost>
- [4] Kaggle. Predict future sales. Online, 2020. URL <https://www.kaggle.com/c/competitive-data-science-predict-future-sales>.
- [5] NeurIPS. Formatting instructions for neurips 2019. Online, 2019. URL <https://nips.cc/Conferences/2016/PaperInformation/StyleFiles>.
- [6] Towards Data Science. CatBoost vs. Light GBM vs. XGBoost. Online, 2018. URL <https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db>
- [7] Zhihu. How to make an e-commerce sales forecast model. Online, 2019. URL <https://www.zhihu.com/question/38988348>